The Operating System Handbook

or, Fake Your Way Through Minis and Mainframes

by Bob DuCharme

VMS

Table of Contents

This Section, the Rest of the Book

This is one part of the book "Operating Systems Handbook (or, Fake Your Way Through Minis and Mainframes)," which was originally published by McGraw-Hill as a $49.60 hardcover. Once they reverted the rights to me after it went out of print, I converted the original XyWrite files to DocBook XML and then used Norm Walsh's stylesheets (see www.nwalsh.com) and the Apache FOP program (see xml.apache.org) to convert that to Acrobat files. The six parts of the book are all available at http://www.snee.com/bob/opsys.html:

- Part 1: Introduction. Note that this part includes a new section explaining why I didn't update the book. I strongly suggest that, no matter how much or how little of the book you can use, you glance through the whole Introduction as well. The "Comments and Suggestions" part is now obsolete; my home page is now http:/www.snee.com/bob and my e-mail address is bob@snee.com.
- Part 2: UNIX. Everything described here should apply to Linux and its relatives.
- Part 3: OpenVMS. Basically, VMS. DEC was calling it "OpenVMS" at the time.
- Part 4: OS/400. The operating system for IBM's AS/400.
- Part 5: VM/CMS. An IBM mainframe operating system.
- Part 6: MVS. Another IBM mainframe operating system.

See www.snee.com/bob for information on books I've written since. In reverse chronological order, they are:

- *XSLT Quickly* is a tutorial and users guide to XSLT designed to get you writing stylesheets as quickly as possible.
- *XML: The Annotated Specification* is a copy of the official W3C XML specification with examples, terminology, and explanations of any SGML and computer science concepts necessary for a complete understanding of the XML spec.
- *SGML CD* is a users guide to free SGML software, most of which can be used with XML as well. The chapter on Emacs and PSGML, which requires no previous knowledge of Emacs, is available on the web site in English, Russian, and Polish.

One more thing: either I couldn't figure out the XSL `keep-together` property or version 0.18.1 of FOP doesn't implement it yet. Either way I apologize that some screen shots get split across page breaks.

Chapter 7 OpenVMS: An Introduction

VMS has traditionally been the operating system used on Digital Equipment Corporation's VAX line of computers. OpenVMS is the latest version of VMS, designed to spread its availability beyond the VAX. (The OpenVMS interface has remained similar enough to that of VMS that everything described in these chapters works the same way with recent versions of OpenVMS as they work with VMS. When I refer to "VMS," take it to mean VMS and OpenVMS together.) Today, the VAX is still the primary platform on which OpenVMS runs, and because of the close connection between VMS and the VAX for over fifteen years, understanding the advantages of the VAX makes it easier to understand OpenVMS.

While VAX models range from the VAXstation VLC, which is inexpensive enough to compete in the marketplace with PCs, to the VAX 9000, which has enough power to compete with mainframes, all VAXes (or "VAXen"—see the buzzword sidebar on this) are generally considered to be minicomputers.

One of the VAX's major selling points has always been that, despite the wide range in the size and power of the different models, all models have the same architecture (the hardware design, as it appears to the system software) and all can run the same operating system: VMS. The saying "one architecture, one operating system" often comes up when DEC people discuss VAXes. This means that you can more easily port your applications and data if you outgrow one VAX and get a bigger one. It also makes communication between VAXes easier; if you outgrow one, you don't necessarily have to trade up—you can get a new one and hook it up to the old one, setting them up to work together.

VAXes work well together because they can operate in a "peer-to-peer" relationship. This goes a step beyond client/server computing: in a peer-to-peer system, any computer can be a client and any can be a server. This allows greater flexibility when multiple computers carry out tasks in cooperation. (For more on client/server systems, see the introduction at the beginning of this book.)

It's not difficult to connect several VAXes together so that they work as a cooperating unit called a "VAXcluster." In fact, the "one architecture, one operating system" approach to the design of the VAX line makes this easier with VAXes than with just about any other computer. (Following the VAX's lead, most mini and PC manufacturers are working hard to catch up.) This cooperation is another of the VAX's advantages: it makes it easier to share resources like printers and disk drives and it reduces the chance of problems if part of the system malfunctions. So the VAX you use—especially if it's located at a university or company large enough to own several—may very well be part of a VAXcluster.

DEC developed VMS and the VAX together. One benefit of this combination is the idea of "balanced architecture," or how well the designers balanced the tradeoffs between work done by the hardware and work done by the operating system. For computers running an operating system like UNIX, where programs are supposed to be portable from one company's UNIX machine to another's, designers must engineer the hardware to accommodate the lowest common denomina-

tor of the various operating system versions. This shifts much of the burden of management in a complex system to the operating system software. Because VMS and the VAX architecture were designed together, DEC engineers could make tradeoffs with fewer compromises. VMS fans assert that this cooperation between the hardware and the system software makes for a much more efficient computer.

Perhaps the greatest reason for the popularity of VMS is its ease of use. The commands are English words that are easy to remember, and the on-line help, while extensive, is simple to figure out. Utilities included with the system, like the mail program and text editors, are also powerful without intimidating beginners, and feature their own very good on-line help.

7.1 History

DEC released the first VAX, the VAX-11/780, in 1977. Its power and quality sent shock waves through the minicomputer industry. (Tracy Kidder's book "The Soul of a New Machine" describes Data General's feverish response to the VAX, as they built a new mini to compete with it. I highly recommend this classic, the only book on computers that I know of to win a Pulitzer prize.)

In 1977, the only computer company bigger than DEC was IBM. If you outgrew a smaller IBM system, a lot of work, time, and expense was necessary to move your programs and data to one of their bigger systems. This made the VAX's flexibility a strong selling point to organizations that needed a minicomputer and aspired to more power further down the road. Academic institutions found them particularly appealing, and today VAXes are very common at many universities.

VMS versus UNIX

Even if no one had developed versions of UNIX to run on VAXes, proponents of UNIX and VMS would still be engaged in a long, loud debate over the relative merits of the two operating systems. (The VAX was actually the target for one of Bell Labs' first ports of UNIX. Because this version didn't take advantage of the VAX's virtual memory capability—the "VM" in "VMS"—developers at the University of California at Berkeley developed a new version of UNIX that did. The success of Berkeley UNIX made it the primary alternative to AT&T's UNIX, and it became the standard in academic environments.) In Clifford Stoll's book "The Cuckoo's Egg," he describes how the VMS and UNIX system administrators at the computer center where he worked loved to disparage each other's operating systems. Their accusations and jibes reveal much

about how the two main camps in the minicomputer world view each other.

The rivalry between UNIX and VMS can be intense because they compete for much of the same turf. Small companies looking for something to keep track of their inventories and payrolls might use an IBM minicomputer, but academic, medical, scientific, and engineering organizations often have little interest in a "business machine" (the "BM" in "IBM"). They want a computer that gives them power without much cost and the flexibility to let them design their own software.

VMS and UNIX both offer this, but with different approaches. Proponents of VMS claim that UNIX is difficult because its abbreviated commands are cryptic and difficult to learn. They view the UNIX approach of offering you tools and ways to combine them as a disadvantage, because assembling a collection of pieces—especially when they come from different companies—can frustrate all but experienced experts. Compared to UNIX, VMS is easy to learn, and it's easier to coordinate a large amount of VAX hardware and software.

Proponents of UNIX sometimes accuse VMS of being slow, but the wide range of VAXes available mean that there's always a faster or slower model available. They also turn around some of UNIX's alleged disadvantages and call them advantages: they claim that the tools approach gives greater flexibility in putting together the system that they really need, and the wide variety of vendors involved eliminates dependence on the whims and fortunes of a single large corporation—in this case, DEC.

Regardless of individual opinions, VMS and UNIX coexist at many companies and universities. Being comfortable with both is always better than familiarity with one and contempt for the other, because the increasing ease of communication between the two operating systems means that more and more sites are hooking them up into one big cooperating network.

7.1.1 Today

A 1992 estimate put the number of VAX/VMS systems in the field at 500,000 and the number of users at 10 million. OpenVMS may outlive the VAX, or at least the VAX as we know it; DEC's new Alpha processor, a significant jump from the VAX, lies at the heart of their next round of technology. It's based on RISC technology, a new method for designing processors that is playing an increasingly larger role in many companies' new hardware. Although DEC designed the Alpha chip to accommodate several different operating systems, they made sure that OpenVMS would run well on it. Rest assured that OpenVMS will be with us for a long time.

7.1.1.1 Popular VMS Software

You can choose from a wide variety of complete software packages available to run under VMS. By "complete," I mean to distinguish it from UNIX—the UNIX philosophy encourages a wide choice of specialized tools that one can piece together into customized applications more than it encourages the kind of complete software packages familiar to PC users. If you have to take an office full of people, set them up with a multi-user computer, and train them in the operating system and the applications they need as quickly as possible, you will probably want to give them either a system running VMS or an AS/400. Of these two, VMS's seniority means that you have a greater range of complete software packages to choose from, so there's a better chance that the software you need already exists and is optimized to take advantage of the VAX. (AS/400s can run programs from IBM's System/36 and System/38 lines, but these don't take advantage of the AS/400's architecture as well as custom-written applications do.)

One big-selling VMS package is DEC's ALL-IN-1 office management system. It provides word processing, electronic mail, meeting and appointment scheduling, and other features that make it popular at many VMS sites.

Another DEC software product closely associated with VMS is Rdb, a relational database system. Although relational, Rdb does not use SQL (Structured Query Language) to manipulate data; in order to avoid this IBM creation DEC came up with their own system, called Digital Standard Relational Interface (DSRI—the "Standard" part is somewhat ironic). To develop Rdb applications, you must purchase software from DEC, but to run them, you only need RDO (Relational Database Operator), which is included with VMS. This policy has made Rdb popular in the VMS world, and other database systems must coexist efficiently with it in order to compete in the VMS marketplace.

Pathworks consists of a series of programs from DEC that allow, among other things, a system running VMS to act as a file and print server on a network with DOS, OS/2, or Macintosh microcomputers as clients. As the minicomputer's role changes from being *the* computer in a particular company or department to *a* computer cooperating with smaller ones, Pathworks and Novell products like NetWare for VMS make VMS systems better suited to taking on file and print server roles than any other popular minicomputer.

7.1.2 VMS, DCL

VMS stands for "Virtual Memory System." The name derives from the technique it uses to manage memory: the use of virtual memory is the ability to give programs access to more memory than the computer actually has.

VMS users refer to commands as "DCL commands" rather than "VMS commands." DCL, or "Digital Command Language," is the language developed by DEC to tell VMS what to do.

Every command-line driven operating system has a language (really, just a collection of commands) for us to tell it what to do; VMS is the only operating system that I know of that has a separate name for that language. DCL commands are not that different from the commands on other major operating systems (for example, `COPY`, `RENAME`, and `DELETE`). What confuses beginners is the existence of a separate name to refer to the command language. To some, it implies the existence of special commands above and beyond the "normal" commands used to communicate with the operating system, like the relationship of REXX to the EXEC language on IBM mainframes.

We refer to VMS command files as "DCL command procedures" because, like the command files in other operating systems, they let you string together operating system commands in a text file and then run the series of commands by typing that file's name at the command prompt. Section 11.2, "Command Files," covers this in greater detail.

> **BUZZWORD** *Vaxen* The official DEC plural for VAX computers is just that: VAX computers. "VAX," a trademarked brand name, is an adjective that modifies "computer" the same way that "Xerox" is legally a trademark to be used as an adjective modifying the word "photocopier." So just as you're not supposed to say "make some copies with the Xerox," you're not supposed to refer to the computer as a "VAX" or the plural as "VAXes."
>
> People say "VAXes" anyway. But another plural form has arisen: "VAXen." It seems to be preferred in more academic, less corporate environments. It's difficult to trace its etymology. The resemblance to the word "oxen" brings to mind a popular story that compares DEC's computer to the animal. DEC people like to point out that when a farmer realizes that his holdings have grown to the point where his one ox is no longer enough to plow the fields, he does not look for a bigger ox to replace his original one; instead he purchases an additional ox—or maybe several—and yokes them together to combine their power. The farmer increases his plowing power without giving up his original source of

power. The connectivity (to use an overly popular industry buzzword) of VAX computers makes oxen a good metaphor for their strengths.

In any case, "VAXes" and "VAXen" mean the same thing. The latter is, at most, probably considered slightly hipper.

Chapter 8 Getting Started with OpenVMS

8.1 Starting Up

When you turn on a terminal connected to a VMS system, or successfully connect to such a system over a network or phone line, the first thing you see is a message that asks for your user ID (or your "username"). This might be preceded by a message telling you the system you've connected to:

```
          WELCOME TO THE NEPTUNE VAX SYSTEM
             UNAUTHORIZED USE PROHIBITED
                O'ROURKE ENTERPRISES
Username:
```

As an authorized user of this system you should have a login name that represents your identity on the system. Type it at the `Username:` prompt and press the Return key. If you make a typing mistake, use your Delete key to back up your cursor and erase the previous character. (The use of the Delete key to perform what many regard as the Backspace key's function can be confusing; see section 8.1.2, "Entering Commands," for more on this.)

VMS is not case-sensitive about your username or password. Whether you enter your username in upper or lower case, it appears in upper case. The next prompt asks you for your password:

```
Password:
```

Type it in and press the Return key. The characters of your password should not show up as you type them.

If all went well, you will be logged in. The system may display some messages from the system administrator before it gets to the DCL command prompt, which is usually a dollar sign.

8.1.1 Finishing Your VMS Session

To log out, type the `logout` command at the DCL `$` prompt. The following shows the system's response when Joe User enters this command:

```
$ logout
JOEUSER     logged out at 12-JUL-1994 14:01:58.30
```

8.1.1.1 Reconnecting

If you are ever accidentally disconnected from your session without properly logging out, log back in as you normally do. Some systems will display a message like this:

```
    You have the following disconnected process:
Terminal   Process name    Image name
VTA1411:   JOE USER        $1NEPDISK:[SYS4.SYSCOMMON.][SYSCBI]VMS$INT.EXE;2
```

```
Connect to above listed process [YES]:
```

This tells you that you left off in the middle of something the last time you were connected, and it asks if you want to resume where you left off. The square braces around the word "YES" show that it is the default; if you press Return, it assumes that you mean "YES" and the system resumes whatever you were doing just before you were disconnected. A response of "NO" tells the system to put you at the DCL $ prompt.

Not all VMS systems have this feature; some display the $ prompt as if you were logging in normally when you reconnect after an aborted session.

8.1.2 Entering Commands

When you enter a command at the DCL $ prompt and press Return, the output appears under it and the screen scrolls up if necessary to show the output. You only need to type the first four letters of any DCL command, because no two DCL commands begin with the same first four letters. If a command does not have the same first three letters as any other DCL command, which is often the case, you can get away with only using the first three letters.

When using a microcomputer to emulate a terminal, the most popular terminals to emulate are those of DEC's VT (for "Virtual Terminal") series. Nearly all telecommunications programs can emulate the VT100, one of the original models in this series, and nearly all minicomputers and mainframes can work with a VT100. Most telecommunications programs and computers can also work with more advanced VT terminals, which are named with higher numbers like VT220 or VT340.

Users of some keyboards on other computers, especially IBM-based ones, are accustomed to a Delete key that deletes the character at the cursor and a Backspace key that deletes the character to the left of the cursor. VT keyboards are more like the Macintosh's (actually, the Mac keyboard is more like the VT's): the key known as the "Delete" key (in the upper-right of the keyboard, with a pentagon pointing to the left and an "X" inside the pentagon) deletes the character just before the cursor position. The VT keyboard has no specific key for deleting the character at the cursor position. Terminal emulation programs often assign a PC's Delete key to "delete" in the VT sense of the word, so PC users must get used to using their Delete key to perform a function that they are accustomed to doing with their Backspace key.

To sum up, PC users who want to correct a typing mistake when connected to a VMS system must get used to pressing their Delete key to get their cursor back to the place where they made a mistake. (To confuse things a little more, note that I said "when connected to a VMS system"—when emulating a VT terminal while connected to an IBM or UNIX machine, Delete usually deletes the character at the cursor and Backspace deletes the character on the cursor's left.)

One more thing about a very important key: although certain IBM mainframe keyboards have separate Enter and Return keys, on most other computer keyboards one or the other serves the

purpose of both. VT keyboards only have a Return key, which is what you press when you've finished typing a command and want to execute it. On a PC emulating a VT terminal, the Enter key stands in for the Return key. Either way, it's the big one above your right-shift key.

8.1.2.1 Retrieving Previous Commands

Pressing Ctrl+B retrieves previously entered commands, one at a time, to the command line. If you receive an error message because of a typo in a command you just entered, this saves you some typing, because you can retrieve the command in one keystroke and fix your mistake instead of retyping the whole command. (With some terminal emulation setups, the Cursor Up key also does this.)

8.1.2.2 Aborting Screen Output

Ctrl+Y is known as the "Interrupt" key. If a command is displaying screens and screens of output and you don't want to see any more, press Ctrl+Y. VMS displays the word "Interrupt" to show where you stopped the output and it puts the DCL command prompt underneath it, ready for your next command.

Ctrl+Y can interrupt more than just screen output. If things get out of hand with just about any VMS program, it's one of the first things to try if you want to abort your session's current activity.

If you press Ctrl+Y by mistake, enter `CONTINUE` at the DCL command prompt to resume the running process.

8.1.2.3 Command Parameters

Many commands need some information from you in order to do their job. For instance, when you type the `COPY` command, the system needs to know the name of the file you want to copy and the name you want to assign to your new copy. As you'll see in the section on the `COPY` command, you could type this:

```
copy filename.old filename.new
```

If you type

```
copy
```

by itself, VMS responds in a fairly user-friendly way to this abbreviated syntax—it prompts you for the information it needs. At the `_From:` and `_To:` prompts that appear, you indicate the names of your source and destination files. For example:

```
$ copy
_From: inven.c
_To: inven_c.bkp
```

```
$
```

8.1.2.4 Case Sensitivity

VMS is not case-sensitive. Whether you enter your commands in upper or lower case, they still have the same effect. The same applies to file names: when you create or refer to a file, whether you write out its name in upper or lower case, VMS translates it to upper case.

8.2 File Names

VMS users loosely use the term "file name" to refer to the two-part name that identifies a specific file in a directory. I say "loosely" because, more formally, it refers to the first of these two parts, with the second part being the file type. Both the file name and file type can be up to 39 characters long, but the most commonly used file types are three letters long.

Like its counterparts on other operating systems, the file type is like a person's last name. It identifies the family to which the file belongs. For example, the following file types indicate that a file is a member of the following families:

`EXE`	An executable program.
`BAS`	A program written in the BASIC programming language.
`LIS`	A plain text file.
`DIR`	A special file type: a `DIR`, for all practical purposes, is a subdirectory, or subdivision, of the current default directory. Technically, it too is a file, but a file that keeps track of the files in the subdirectory that it names. If you try to look at this file, it will mostly look like gibberish. For more on subdirectories, see section 8.3, "How Files Are Organized."

You can use letters of the alphabet, numbers, and underscores in file names and types. Spaces are not allowed. You'll be pushing your luck if you use any other characters—if a command doesn't work and VMS gives you a message like "check use of special characters," you may have tried to create a file with a carat (^), a pound sign (#), or one of the keyboard's other non-alphanumeric characters.

VMS keeps multiple versions of each of your files. The three versions of `memos.lis` might be called `memos.lis;1`, `memos.lis;2` and `memos.lis;3`. The one with the highest number is the most recent version. If you refer to a file but don't include its version number, many commands assume that you're referring to the most recent one, but some commands (like `DELETE`) force you to include the version number when you tell it which file or files to act on. Section 8.3,

"How Files Are Organized," covers version numbers in more detail.

8.2.1 Wildcards

The main wildcards in VMS are the asterisk and the percent sign. Although the examples below demonstrate their use with the `DIRECTORY` command, which lists file names, remember that you can use them with almost any command that uses file names as command-line parameters. For more information, see the material on wildcards in section 1.5, "General Advice."

8.2.1.1 The Asterisk

An asterisk at the end of a file name means the same thing in VMS that it means in most other operating systems. It can represent zero or more characters at that position in the file name or file type.

In VMS, you're not restricted to putting the asterisk at the end of the file name or file type; if you use the `DIRECTORY` command to list file names and enter the following,

```
DIRECTORY *MAY.LIS
```

the system lists the names of files with a file type of `LIS` and a file name that ends with the letters "MAY." (Because the asterisk represents zero or more characters, the file `MAY.LIS` fits that pattern as well as `CAMAY.LIS` would.)

Entering this

```
DIRECTORY MAY*94.LIS
```

lists files with "LIS" as a file type and a file name that begins with "MAY" and ends with "94" whether there are 0 or 34 characters between the "MAY" and the "94." (Remember, the file name and type can each be a total of 39 characters, and "MAY" and "94" make five.)

To list file names that have "MAY" anywhere in the file name, regardless of the file type, enter

```
DIRECTORY *MAY*.*
```

If you don't include any characters or wildcards to indicate a pattern for the file type, VMS assumes a default of `*`. In other words, it includes files with file names that fit the entered pattern regardless of the file type. For example, entering

```
DIRECTORY *MAY
```

lists all files with a file name that ends with "MAY," no matter what their file type.

8.2.1.2 The Percent Sign

The percent sign represents a single character in a file name—no more, no less. Several percent

signs represent that number of characters, so that

```
DIRECTORY %%%93RPT.TXT
```

would list out the file names with exactly three characters before the characters "93rpt.txt."

8.3 How Files Are Organized

As with several other operating systems, files on a VMS system are organized in divisions and subdivisions of hard disk space called directories, and a file's full name (called its "file specification," or "filespec") includes the directory name. Unlike other operating systems, the complete filespec includes more than the directory name: it also includes the names of the computer and hard disk on which the file is stored.

The following shows a possible complete filespec for Joe User's file `FOR_MARY.LIS`:

```
NEPTUNE::NEPDISK:[JOEUSER.MEMOS.JUNE]FOR_MARY.LIS;3
```

The filespec consists of the following parts:

`node` — The name of the computer on which the file is stored. This is not a brand name or model name, but more of a nickname that system administrators assign when they install a computer. If you send electronic mail to someone using a VMS system other than your own, you need to know their node name. In the example, Joe User's system administrator named their node `NEPTUNE`. Note the use of the two colons to separate the node name from the rest of the filespec.

`device` — A given VMS system may store files on several hard disks; the device name is the specific hard disk where a file is stored. `NEPDISK` is the device name in the example. A single colon separates the device name from the rest of the file name. (DOS users will find this use of the colon to separate a disk name from a file and directory name familiar—for example, in `c:command.com` the disk name is `c` and the file name is `command.com`.)

`directory` — A hard disk is divided into sections called directories. The system administrator assigns each user ID its own directory, and the user can create subdivisions of this directory known as subdirectories. (These too may be subdivided, and the subdivisions may be subdivided, and so forth to seven levels below the user ID's directory.) A directory's full name consists of the names of the various levels separated by periods with square braces around the whole thing.

We call a user ID's main directory—in the example above, `[JOEUSER]`—that user's "root directory." DOS and UNIX users should keep in mind that in VMS "root directory" means the root of a particular user's arrangement of subdirectories (like a UNIX user's home directory), rather than the root of the whole hard disk.

Joe User might divide his `JOEUSER` directory into several subdirectories and name one of them `MEMOS`. This subdirectory's full name would be `[JOEUSER.MEMOS]`. If he divided his `MEMOS` subdirectory into subsections for each month of the year, the full name of the subdirectory that holds his June memos would be `[JOEUSER.MEMOS.JUNE]`, as in the example.

`file name` — This means just what it says. See section 8.2, "File Names," for information on the two parts of a file name. In the example, the file name is `FOR_MARY.LIS`.

`version number` — When you first create a file, VMS assigns it a version number of 1. If you edit the file, the text editor saves the edited version with a version number of 2 and the old one remains in the same directory with a version number of 1. The version number is appended onto the file name, separated from it by a semicolon. In the example, the `;3` after the file name `FOR_MARY.LIS` shows that the filespec represents the third version of the file.

You will undoubtedly find that your directories gradually fill up with unnecessary old versions of your files. Section 9.1.7.1, "Purging Old Versions of Your Files," explains how to delete all but the most recent version or versions of a file.

The system administrator assigns your user ID to a particular disk of a particular computer. If you don't include a node or device name when you refer to a file, VMS assumes that you're talking about a file on the same node and disk that you're already using.

Why does VMS give you the option of including the computer and hard disk name in a file's complete name, when few other operating systems do? As I mentioned earlier, VAXes are often grouped together into a network called a VAXcluster. The ability to be so specific about a file's location makes it much easier to access a file on other VAXes in your cluster.

8.3.1 Moving Between Directories

The command `SHOW DEFAULT` displays the name of your "default" directory. If Joe User types this while his root directory is the default, the system displays the following:

```
NEPDISK:[JOEUSER]
```

While you get used to the various syntax possibilities for changing your default directory, this command is very handy, because you can use it to check the success of each attempt to make a new directory the default one.

The SET DEFAULT command changes your default directory. Because this command and SHOW DEFAULT come up so often, it saves you some typing to remember that you only need the first three letters of any DCL keywords. This makes these commands SET DEF and SHOW DEF—or even SHO DEF.

For Joe User to set his default directory to the MEMOS subdirectory of his root directory, he could type this:

```
SET DEFAULT [JOEUSER.MEMOS]
```

Fortunately, there are other shortcuts besides abbreviating the keywords. If you begin the directory name with a period, you tell VMS that the following directory name is a subdirectory of the current default one. For example, if Joe is already in the [JOEUSER] directory and wants to go one level down to the MEMOS directory, he can type this:

```
SET DEFAULT [.MEMOS]
```

If he wanted to go from [JOEUSER] to [JOEUSER.MEMOS.JUNE] in one command without typing the entire name of his destination, he could enter this:

```
SET DEFAULT [.MEMOS.JUNE]
```

When moving back one or more levels, a hyphen represents the directory one level closer to the root. For example, Joe could go from [JOEUSER.MEMOS.JUNE] to [JOEUSER.MEMOS] by entering this:

```
SET DEFAULT [-]
```

To go back multiple levels, you can use multiple hyphens, as long as you separate them with periods. For example, Joe can go from [JOEUSER.MEMOS.JUNE] to his [JOEUSER] root directory by entering this:

```
SET DEFAULT [-.-]
```

As you'll see in section 9.1.5, "Copying Files," these tricks for abbreviating directory names also work when you specify a directory other than the current default one as the destination for a file that you are copying.

8.3.1.1 Default Directory? Current Directory?

There is a subtle difference between the VMS concept of a default directory and the DOS or UNIX concept of a "current" directory. When you make a new directory current while using one

of the latter two operating systems, you can think of it as moving to a new part of the disk to work. When you set a default directory in VMS, you indicate a directory (or disk) name to use with any files for which you don't explicitly include the directory or disk name. Misunderstanding this difference can lead to aggravation when you get certain error messages.

For example, this happens when you specify a non-existent directory as your new default directory. In UNIX and DOS, when you issue the `cd` (change directory) command with a non-existent directory as its parameter, the system gives you an error message right away. On the other hand, when you use the `SET DEFAULT` command to tell VMS to "apply this directory name to files that I refer to in future commands," the system takes it on faith that the name is a good one. When you try it with a non-existent directory name, the system responds with the DCL prompt, and no error message:

```
$ set default [.febyooary]
$
```

If Joe User tries to do anything with this non-existent directory, he gets an error message or two:

```
$ dir
%DIRECT-E-OPENIN, error opening NEPDISK:[JOEUSER.FEBYOOARY]*.*;* as input
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
```

(See section 9.1.2, "Common Error Messages" for more on the format of error messages.) This problem will continue until you fix it, so Joe types

```
set default [joeuser.february]
```

to correct his mistake.

8.3.2 Querying Available Disk Space

A VMS system administrator allocates a certain amount of disk space to each ID. To see how much of yours you have used and how much free space remains, enter the following command:

```
SHOW QUOTA
```

VMS displays a message similar to this:

```
   User [207,JOEUSER] has 356 blocks used, 9644 available,
   of 10000 authorized and permitted overdraft of 100 blocks on NEPDISK
```

The `[207,JOEUSER]` is the User Identification Code (UIC), a unique code that the system uses to keep track of users. A block represents 512 bytes, so to picture a given number of blocks in kilobytes, cut it in half. The preceding example shows that the files in Joe User's ID space take up about 178 K, and he has about 4822 K of free space. Section 9.1.3, "Listing File Names," shows how adding the `/SIZE` qualifier to the `DIRECTORY` command displays the size, in blocks, of individual files.

8.4 Available On-line Help

VMS probably has the best on-line help of any operating system that uses character-based screens. In fact, it outshines the help included with many operating systems that use a graphical user interface. Its greatest strength is the ease with which you can move to a greater or lesser level of detail and from topic to topic.

If you type HELP by itself at the DCL prompt, VMS displays an introductory help screen similar to the one in Figure 8.1.

```
HELP

   The   HELP   command   invokes   the  VMS  HELP  Facility  to  display
   information about a VMS command or topic.  In response to the "Topic?"
   prompt, you can:

      o Type  the  name of the command  or topic for which you need help.

      o Type  INSTRUCTIONS  for more detailed  instructions on how to use
        HELP.

      o Type  HINTS  if you are not  sure of the  name  of the command or
        topic for which you need help.

      o Type a question mark (?) to redisplay the most recently requested
        text.

      o Press the RETURN key one or more times to exit from HELP.


Press RETURN to continue ...
```

Figure 8.1 Screen 1 of the VMS introductory help screen.

This first help screen tells you how to use help. Press Return, and the next help screen looks similar to the one shown in Figure 8.2.

```
   You  can  abbreviate any  topic name, although ambiguous abbreviations
   result in all matches being displayed.


  Additional information available:
```

```
   :=          =           @           ACCOUNTING ALLIN1     ALLOCATE   ANALYZE
   APPEND      ASSIGN      ASU         ATTACH     AUTHORIZE  AUTOGEN    BACKUP
   BASIC       BTEQ        CACHE       CALL       CANCEL     CC         CDD
   CDDL        CDDV        CDD_PLUS    CDO        CLOSE      CMS        COBOL
   COLLECT     CONNECT     CONTINUE    CONVERT    COPY       CREATE     DATATRIEVE
   DBCCP       DBC_1012    DBMS        DBO        DBO40      DDL        DDL40
   DEALLOCATE  DEASSIGN    DEBUG       DECK       DECtrace   DEFINE     DELETE
   DEPOSIT     DICTIONARY  DIFFERENCES            DIRECTORY  DISCONNECT DISKQUOTA
   DISMOUNT    DML         DML40       DMU        DOCUMENT   DSM        DTM
   DUMP        EDIT        ENCRYPT     EOD        EOJ        EXAMINE    EXCHANGE
   EXIT        FDL         FileChief   FINGER     FMS        FONT       FORTRAN
   FTP         GOSUB       GOTO        GRAPH      HELP       Hints      IDL
   IDMCOPY     IDMDATE     IDMDUMP     IDMFCOPY   IDMLOAD    INITIALIZE INQUIRE

 Press RETURN to continue ...
```

Figure 8.2 Screen 2 of the VMS introductory help screen.

This shows you the beginning of the list of available help topics. (Note that `Hints` is on the list. The first help screen already indicated that this is a valid help topic.)

Your screen probably won't look exactly like this; part of the beauty of VMS on-line help is the ease with which system administrators can add new help topics, so that users can learn about issues that pertain to their particular system just as easily as they can learn about VMS commands.

When you press Return again, VMS displays the rest of the list, and then a prompt that asks you which topic you want to learn more about:

```
Topic?
```

If you respond by merely pressing the Return key, VMS returns you to the DCL prompt. If you want to know more about one of the listed topics, enter its name. For example, if you respond to the `Topic?` prompt by entering `EDIT`, the help program displays a screen similar to the one shown in Figure 8.3.

```
EDIT

    The EDIT commands perform the following functions:

    o  Invoke the Access Control List Editor  to  create  or  modify  an
       access control list for an object (see /ACL).

    o  Invoke the EDT screen-oriented editor (see /EDT).
```

```
    o  Invoke the FDL  editor  to  create  and  modify  File  Definition
       Language files (see /FDL).

    o  Invoke the SUMSLP batch-oriented editor to update a single  input
       file with multiple files of edit commands (see /SUM).

    o  Invoke the TECO editor (see /TECO).

    o  Invoke the TPU editor (see /TPU).

Press RETURN to continue ...
```

Figure 8.3 VMS help screen 1 for the EDIT topic.

Pressing Return another time displays the second and final general help screen for the `EDIT` command, as shown in Figure 8.4.

```
  Additional information available:

  /ACL       /EDT       /FDL       /SUM       /TECO      /TPU

EDIT Subtopic?
```

Figure 8.4 Additional information available on the EDIT command.

Note how `EDIT`, as a topic, has subtopics that you can learn more about. Some subtopics have their own subtopics; if you enter `/TPU` in response to the `EDIT Subtopic?` prompt, VMS displays introductory information about the Text Processing Utility, and then another subtopic menu to select from, as shown in Figure 8.5.

```
    Additional information available:

    EVE_Editor Examples   Logicals   Parameter  Qualifiers /COMMAND   /CREATE
    /DEBUG     /DISPLAY   /INITIALIZATION       /INTERFACE /JOURNAL   /MODIFY
    /OUTPUT    /READ_ONLY /RECOVER   /SECTION   /START_POSITION       /WORK
    /WRITE
```

```
EDIT /TPU Subtopic?
```

Figure 8.5 VMS help screen 2 for the EDIT topic.

If you respond to any help prompt by pressing Return without first entering anything, the help program takes you back to the previous help level's prompt. In the following, note what happens when a user looking at the `EDIT /TPU Subtopic?` prompt repeatedly presses Return until the system returns to the DCL prompt:

```
EDIT /TPU Subtopic?
EDIT Subtopic?
Topic?
$
```

When you reach the deepest level of help available on a particular topic (that is, when the currently displayed topic has no subtopics), the help program redisplays the most recent help prompt. For example, if `EDIT /TPU` has no subtopics of its own, the system displays the `EDIT Subtopic?` prompt after it displays the help information on `EDIT /TPU`.

What if you respond to a help prompt with something that the help program doesn't recognize, like `POTRZEBIE`? The help program responds with a polite message like this:

```
Sorry, no documentation on POTRZEBIE
```

Note that one of the `EDIT /TPU` subtopics is `EVE_Editor`. Topic and subtopic names cannot have spaces in them, so they use underscores instead. For example, one subtopic of the `HINTS` screen is called `Batch_and_print_jobs`. Don't forget to include the underscores when you enter one of these topic names, or you'll get the "Sorry, no documentation" message.

You don't always have to go through the various menus and screens to get help. If you need help with a specific command, enter `HELP` followed by the command name at the DCL prompt. For example, to go right from the DCL prompt to the main help screen for the `EDIT` command, enter this:

```
HELP EDIT
```

For even more specific help, you can add one of a command's qualifiers to go to the help screen for that qualifier. For example, entering

```
HELP EDIT/TPU
```

takes you right to the screen that describes the TPU editor.

As you can see, the VMS help system is easy to get into, easy to navigate, and easy to get out of. Also, as the very first screen shows, you can easily find out more about the help program from the help program itself.

Chapter 9 Using Files in OpenVMS

9.1 The Eight Most Important Commands

The eight most important commands in VMS are:

`DIRECTORY`	lists file names.
`TYPE`	displays the contents of files.
`COPY`	copies files.
`RENAME`	renames files.
`DELETE`	deletes files and directories.
`PURGE`	erases old versions of your files and keeps recent ones.
`SET PROTECTION`	grants and revokes access to files.
`CREATE/DIRECTORY`	creates directories.

9.1.1 Command Options: Qualifiers

A command option, known as a "qualifier" in VMS, is separated from the command by a slash. Although any spaces before or after the slash don't affect the command's execution, you usually see examples written with no spaces before or after the slash. VMS users usually consider the slash to be part of the qualifier, not a separator between qualifiers. For example, when you're viewing the on-line help's basic description of the `EDIT` command and want to see subtopic information on its `/TPU` qualifier, entering just "TPU" as the subtopic name displays an error message. You must include the slash for VMS to recognize what you want.

The `DIRECTORY` command entered by itself only lists file names. Various qualifiers allow you to list more information with these file names. For example, the `/SIZE` qualifier lists each file's size, in blocks, next to its name, regardless of whether you enter

```
DIRECTORY/SIZE
```

or

```
DIRECTORY /SIZE
```

or this:

```
DIRECTORY/ SIZE
```

This latitude with spaces still applies when you add more than one qualifier to your command. For example, adding the `/DATE` qualifier tells the `DIRECTORY` command to list the date of the file's creation, along with the file's name. To list each file's size and creation date, you could enter

```
DIRECTORY/DATE/SIZE
```

or

```
DIRECTORY / DATE / SIZE
```

without any problem.

9.1.2 Common Error Messages

VMS error messages take the following form:

```
%component-c-abbrev, message
```

This consists of the following parts (don't worry about the first four—the `message` part is the important one):

`%`	Marks the beginning of the error message. If the message includes more than one line, the additional lines give details about the main error message. These have the same format as the first, but begin with a hyphen (`-`) instead of a percent sign.
`component`	Is the component of VMS that choked on the command that caused the error. This will probably appear as some initials.
`c`	Is the code that tells you the severity of the error:
`I`	Informational
`E`	Error
`S`	Success
`F`	Fatal error
`W`	Warning
`abbrev`	Is a unique error code abbreviation.
`message`	Is the actual error message. With luck it will be a whole sentence that

explains what went wrong.

The most classic error is to misspell a command name so that the system doesn't recognize the entered command. In the following example, Joe User adds an extra "d" to the `DIRECTORY` command:

```
$ ddirectory
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
\DDIRECTORY\
```

VMS first tells him that it doesn't recognize the command he entered. It then repeats, between the slashes, the command that it didn't recognize. This is handy if the system rejects a complicated command and you don't know which part of it caused the error message.

The other classic error is to misspell a file name so that you enter a command to do something to a nonexistent file. For example, let's say Joe User wants to delete the file `SUMRPT.LIS;1` but makes a typo when he enters the file name:

```
DELETE SLUMRPT.LIS;1
```

VMS responds with these messages:

```
%DELETE-W-SEARCHFAIL, error searching for NEPDISK:[JOEUSER]SLUMRPT.LIS;1
-RMS-E-FNF, file not found
```

The message on the first line indicates that there was a problem searching for the `SLUMRPT.LIS` file. The second line elaborates on the problem: it couldn't find the file. (Another possible problem that might cause a searching error would be a hardware problem on the hard drive's disk.)

9.1.3 Listing File Names

The VMS command to list out the files in a directory is `DIRECTORY`. Because this command is used so often, its first three letters are more popular than the final six, especially among DOS users who are used to entering "DIR" on their PCs to list file names.

Entering this command with no qualifiers displays the directory's full name, the file names and version numbers arranged alphabetically in several columns across the screen, and the total count of the file names listed at the bottom, as shown in Figure 9.1.

```
$ dir

Directory NEPDISK:[JOEUSER]

INVEN.C;6           INVEN.C;5           INVEN.C;4           INVEN.EXE;3
INVEN.HLP;1         INVEN.JOU;2         INVEN.JOU;1         INVEN.OBJ;3
L543.TMP;1          L544.TMP;1          L545.TMP;1          MEMOS.DIR;1
MENU.C;4            MENU.C;3            MENU.EXE;2          MENU.EXE;4
MENU.HLP;1          MENU.OBJ;3          SETUP.COM;1         SETUP.DAT;2
```

```
SUMRPT.C;2          SUMRPT.C;1          SUMRPT.EXE;2        SUMRPT.EXE;1
SUMRPT.HLP;1        SUMRPT.LIS;1        SUMRPT.OBJ;2        SUMRPT.OBJ;1
TEST.C;1            TEST.EXE;4          TEST.HLP;1          TEST.OBJ;1
TESTPAS.EXE;1       TESTPAS.PAS;1       TODO.TXT;8          TODO.TXT;7
TODO.TXT;6          TODO.TXT;5          UR.COM;1            URBASE_TEST.DIR;1

Total of 40 files.
```

Figure 9.1 Sample output of DIR (DIRECTORY) command.

If you enter `DIR` without specifying the directory in which to list the files, it lists the files in the default directory. To list the files in another directory, add its name (all shortcuts for directory names are allowed) after the `DIR` command, like this:

```
dir [.memos.june]
```

If you don't indicate a file specification, the `dir` command lists all the files in the directory. If you enter a file name, it only lists versions of that one file. (Note that I said "versions of"; if you include a file name without a semicolon or number after that semicolon, VMS assumes that you want to list all versions of that file.)

It's more common to enter a file specification with wildcards, so that VMS lists a subset of the directory's files. For example, entering the following

```
dir *.c
```

produces output like this:

```
Directory NEPDISK:[JOEUSER]
INVEN.C;6           INVEN.C;5           INVEN.C;4           MENU.C;4
MENU.C;3            SUMRPT.C;2          SUMRPT.C;1          TEST.C;1
Total of 8 files.
```

Adding a file specification to a directory name lists the files in that directory that meet that wildcard pattern. For example, entering

```
dir [joeuser.memos.june]mary*.*
```

lists all the files that begin with the letters "mary" in the `[joeuser.memos.june]` directory.

9.1.3.1 Listing More than File Names

The `DIR` command has many qualifiers; the most useful are `/SIZE`, `/DATE`, and `VERSION=n`.

To learn about the others, enter `HELP DIRECTORY` at the command prompt.

Enter `DIR/SIZE` to list the file names with their individual sizes, in blocks:

```
INVEN.JOU;2              1
INVEN.JOU;1              1
INVEN.OBJ;1              4
L543.TMP;1               2
L544.TMP;1               3
L545.TMP;1               3
MEMOS.DIR;1              1
MENU.C;4                 5
MENU.C;3                 5
MENU.EXE;2               3
```

As explained in section 8.3.2, "Querying Available Disk Space," a block represents 512 bytes, or half a kilobyte. There are no partial blocks; a 513 byte file and a 1024 byte file both take up 2 blocks of space on a VMS system.

The `/DATE` parameter adds the creation date and time, in military format, to each file name. If you are more interested in when a file was last modified than in when it was created, just look at the creation date of the copy of the file with the highest version number. For example, if you enter

```
dir/date inven.c
```

and see this as output,

```
INVEN.C;7             1-AUG-1994 15:20:49.74
INVEN.C;6            30-JUL-1994 16:42:45.85
INVEN.C;5            30-JUL-1994 16:36:35.59
INVEN.C;4            30-JUL-1994 16:24:28.32
```

you can tell that `INVEN.C` was last edited at 3:20 PM on August 1.

If the `DIR` command's output clutters your screen too much, you can instruct it to only list each file's most recent version by adding `/VERSION=1` to the `DIR` command. (`/VERSION=2` tells it to list the most recent two versions, and so forth.) Section 9.1.7.1, "Purging Old Versions of Your Files," shows how to delete older unwanted versions of your files while keeping a specified number of the recent versions.

9.1.4 Displaying a Text File's Contents

To display a text file on your screen, use the VMS `TYPE` command. Followed by a file name, it displays that file's contents. For example, the command

```
type test.c
```

puts the contents of the `test.c` file on the screen:

```
#include <stdio.h>
main()  {
    printf("I hope I can get the C compiler to work.");
```

```
}
```

9.1.4.1 Looking at Text Files One Screen at a Time

Many files are too long to fit on your screen, and the TYPE command scrolls them up your screen too quickly to read. The TYPE command's /PAGE qualifier tells the system to show the file one page at a time. For example, after Joe User enters the following

```
type/page schedule.txt
```

and presses Return, VMS displays the first page of the `schedule.txt` file, as shown in Figure 9.2.

```
October 9
9:00 Ed may have Knicks tickets for me; bug him when he gets back from Toronto
10:30 office supplies sales rep coming
12:00 lunch with Benny postponed until the 10th
2:30 getting teeth cleaned--call 687-2300 first for address
4:00 Fed Ex new diskettes to Chicago

October 10
10:30 meet Dave C., Laurie.  call Laurie first--should I bring new diskettes?
12:30 lunch with Benny
2:00 expecting call from Chicago office.  Have page counts ready.
2:30 Anita's presentation--can I get out of going?
                            Press RETURN to continue
```

Figure 9.2 Sample output of TYPE command with the /PAGE qualifier.

The message `Press RETURN to continue` tells you to press the Return key when you're ready to see the next page. If you find yourself pressing Return over and over and regretting that you issued the TYPE command with such a large file, press Ctrl+Y to abort the display and return to the DCL prompt. (For more on using Ctrl+Y, see section 8.1.2.2, "Aborting Screen Output.")

9.1.5 Copying Files

Copying a file is simple. Enter the `COPY` command followed by the name of the file to copy and the name to assign to the copy. For example, entering the following creates a copy of `inven.c` called `inven_c.bkp`:

```
copy inven.c inven_c.bkp
```

After you press Return, if the system returns you to the DCL prompt with no error message, then you know that VMS copied the file without any problems.

If you enter `COPY` without any parameters, VMS is very forgiving. It prompts you with individual prompts for the information that it needs:

```
$ copy
_From: inven.c
_To: inven_c.bkp
$
```

If you don't include the source file's version number, VMS assumes that you want to copy the most recent version of that file. If you don't include the destination file's version number, VMS starts it at 1 if no file with that name exists. Otherwise, its version number will be one higher than the current highest number assigned to a file of that name.

If you include the destination file's version number, VMS creates the new file with that number. If a file with that name and number already exists, VMS responds with an error message telling you that it couldn't open the destination file for output because it already exists.

You can use wildcards to copy multiple files at once. The following command copies the most recent version of all files with a file name of `mainmenu` to files with a file name of `submenu1`:

```
copy mainmenu.* submenu1.*
```

9.1.5.1 Copying Files to Other Directories

To copy a file to another directory, specify the directory name as the destination of the copy operation. For example, entering

```
copy mainmenu.c [joeuser.inven.old_code]
```

copies the most recent version of the `mainmenu.c` file into the `[joeuser.inven.old_code]` subdirectory. The copy in the destination directory will have the same version number as the source version of the file, even if there are no other files in the destination directory with that file's name.

When you include directory names in the `copy` command's parameters, you can use the same shortcuts to refer to directory names that you can use with the `SET DEFAULT` command to

change the default directory. For example, if `[joeuser.inven.old_code]` is Joe User's default directory, and he wants to copy the `mainmenu.c` program to his root directory—which happens to be the parent of the parent of his default directory—he could type this:

```
copy mainmenu.c [-.-]
```

See section 8.3.1, "Moving Between Directories," for more on abbreviating directory names.

When you copy a file to another directory, the copy will have the same name unless you specify otherwise. Specifying otherwise is easy; just put the new name immediately after the destination directory's name:

```
copy mainmenu.c [-.-]oldmenu.c
```

Be careful to include the brackets when you specify a directory as the destination of your copy operation. For example, if Joe wants to copy the file `080494mj.txt` from his `[JOEUSER]` directory, which is currently his default, to his `[JOEUSER.MEMOS]` directory, he enters this:

```
copy 080494mj.txt [.memos]
```

If he makes the mistake of typing

```
copy 080494mj.txt memos
```

then he creates a new file in his `[JOEUSER]` directory with the name `memos.txt`. (The system assumes that because he did not specify a file type for his new file, he wants it to have the same file type as the source file.)

Here's a trick that won't make sense on other operating systems, but fits right in to the VMS scheme of things. The following is a perfectly valid, almost useful VMS command:

```
copy todo.txt todo.txt
```

In other operating systems, this would mean "make a copy of the `todo.txt` file and call it `todo.txt`"—a command that wouldn't say much. But because of the default version numbers that VMS assigns to the destination of the `COPY` command, VMS reacts to this command by taking the most recent version of the `todo.txt` file and making a copy with the next highest version number.

9.1.6 Renaming Files

To rename a file, enter the `RENAME` command followed by the name of the file to rename and the new name to give it. If you do not include the source file's version number, the most recent version of the file will be renamed. For example, to rename the `inven.c` file as `inven_c.bkp`, enter

```
rename inven.c inven_c.bkp
```

As with the `COPY` command, VMS prompts you for any parameters that you omit.

As we saw in section 8.2, "File Names," directories are really special files, so renaming a directory is as easy as renaming a file if you remember to include the `DIR` file type. For example, if Joe User's current default directory is `[JOEUSER]` and he wants to rename his `[JOEUSER.MEMOS]` subdirectory as `[JOEUSER.LETTERS]`, he enters this:

```
rename memos.dir letters.dir
```

You can also use the `RENAME` command to move a file from one directory to another. For example, entering

```
rename [joeuser.memos]052394js.txt [joeuser.letters]052394js.txt
```

moves the file `052394JS.TXT` from the `[JOEUSER.MEMOS]` directory to the `[JOEUSER.LETTERS]` directory. (In the above example, the second `052394js.txt` is actually unnecessary. Unless you're giving the file a new name in its new location, you don't need to specify its name with the destination directory.)

9.1.7 Deleting Files

To delete a file, enter the `DELETE` command followed by the file name. Unlike other commands, `DELETE` assumes nothing about version numbers when you enter a file name without a version number. Entering `delete` without one, like this,

```
delete inven_c.bkp
```

gives you a message similar to the following:

```
%DELETE-E-DELVER, explicit version number or wild card required
```

The "wild card" part means that if you really want to delete all the versions of `inven_c.bkp`, you can simply enter this:

```
delete inven_c.bkp;*
```

Otherwise, enter a specific version number after the semicolon.

If you are accustomed to another operating system, you'll probably forget to include the version number pretty often when you try to delete a file. This is a great example of how handy the Ctrl+B or Cursor Up keys can be: if you make a simple mistake with your command and get an error message, you can use one of these keystrokes to retrieve the command that you just typed, make the minor correction necessary (in this case, by adding the semicolon and version number), and press Return. See section 8.1.2.1, "Retrieving Previous Commands" for more on this feature.

9.1.7.1 Purging Old Versions of Your Files

The VMS practice of keeping multiple versions of your files can quickly clutter up the disk space allocated to your user ID. VMS has a special command to ease the cleanup of old versions of

your files: PURGE. Entering PURGE with only a file name as a parameter tells VMS to delete all versions of that file except for the most recent one.

To tell VMS to keep more than one version of the file, use the /KEEP=n qualifier. For example, if you have seven versions of todo.txt and enter

```
purge/keep=3 todo.txt
```

VMS deletes the four oldest versions of todo.txt and keeps the three newest ones.

9.1.8 Controlling Access to a File

The SET PROTECTION command lets you control who can read, write, execute, and delete your files and directories. There are four categories of users whose privileges you can set when you set the protection level for a file or files:

System	The system administrators, who have special privileges in order to maintain the system.
Owner	You, because you own the file. (You can't use SET PROTECTION on files that belong to other users.)
Group	For easier system maintenance, system administrators classify users into groups. They might base these groupings on the users' departments or job titles—for example, people in accounting could be one group, and programmers another. When you set a file's protection level, the Group category lets you give your group's members greater access than everyone else on the system.
World	Everyone on the system.

Owners are considered to be members of their own groups, and the owner's group is considered to be part of "World," so if you try to give the group more privileges than the owner, or the world more privileges than the group, it won't work. (It wouldn't make too much sense to try this anyway.)

To find out the protection levels assigned to existing files, add the /PROTECTION qualifier to the DIR command:

```
$ dir/protection ur.com
UR.COM;1             (RWED,RWED,RE,)
```

The commas separate the privilege lists for the System administrators, Owner, Group, and World, in that order. The letters that show the privileges each stand for Read, Write, Edit, and Delete. In the example, the system administrators and owner can perform any of these operations

on the `ur.com` file. Others in the owner's group can read or edit it; the lack of any initials after the final comma shows that people outside the group cannot do anything with it.

To find out the default protection levels assigned to files that you create, enter the command `SHOW PROTECTION`, like this:

```
$ show protection
  SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
```

This makes it pretty clear who has what kind of access to your files, or at least the files to which you don't explicitly assign other access levels.

Use the `SET PROTECTION` command to assign specific access levels. Its parameters, in parentheses, are the relevant user categories and the access that each should have. To assign `ur.com` the access levels shown above, use this command:

```
set protection=(system:rwed,owner:rwed,group:re,world) ur.com
```

Note that `world` is listed with no privileges at all. If you didn't include `world` in the parentheses, you would assign `world` the default access levels shown by the `SHOW PROTECTION` command. In fact, to assign all the default access levels shown by `SHOW PROTECTION`, you only need to type this:

```
set protection ur.com
```

When you assign the different access types that each user category has, you must abbreviate the words Read, Write, Edit, and Delete to their first letters. (No great loss—would you really want to type them out every time?) If you want, you can also abbreviate the user categories to their first letters. This makes the command shown above a bit shorter:

```
set protection=(s:rwed,o:rwed,g:re,w) ur.com
```

It's perfectly OK to use wildcards to specify the files whose access levels you are setting. For example, to make sure that everyone in your group can read but not edit the source code of your existing C programs (which all have a filetype of "c"), and that people outside of your group can't even do that, enter this:

```
set protection=(g:r,w) *.c;*
```

Notice the semicolon and second asterisk after the `c` file type. As with many other commands, if you don't indicate a specific version number of the file that you want `SET PROTECTION` to act on, it only affects the latest revision of the specified file or files. To make sure that it sets the protection level of every single file with a particular file type, remember to add an asterisk as a version number.

9.1.9 Creating Directories

The VMS `CREATE` command can be used to create various kinds of files. Usually, you use the

text editor or an application program to create a file, but you need the CREATE command with its /DIRECTORY qualifier to create subdirectories of your root directory. (Only system administrators can create subdirectories of directories that aren't their own.)

Enter CREATE with the /DIRECTORY qualifier followed by the name of the new directory to create, like this:

```
create/directory [joeuser.memos.july]
```

After you create a new subdirectory, the DIR command will not show any files in it. You still may want to use it to check whether the directory was successfully created. If you use DIR to inquire about the files in a non-existent directory, like this,

```
dir [.potrzebie]
```

VMS responds like this:

```
 %DIRECT-E-OPENIN, error opening NEPDISK:[JOEUSER.POTRZEBIE]*.*;* as input
 -RMS-E-DNF, directory not found
 -SYSTEM-W-NOSUCHFILE, no such file
```

In other words, it couldn't find a potrzebie.dir file in the [JOEUSER] directory to represent a subdirectory of [JOEUSER].

Your subdirectories can have subdirectories and they too can have subdirectories, down to seven levels below your root directory. For example, Joe could create a subdirectory called [JOEUSER.LEV1.LEV2.LEV3.LEV4.LEV5.LEV6.LEV7], but he couldn't create any subdirectories of his LEV7 subdirectory.

9.1.10 Removing Directories

If you know that you can treat a directory called [.whatever] as a file called whatever.dir and you know that you delete files with the DELETE command, then you already know how to delete directories. For example, if Joe User's default directory is [JOEUSER] and he wants to delete the [JOEUSER.URBASE_TEST] directory, he just types this:

```
delete urbase_test.dir;1
```

There is a fairly common problem, however. VMS may respond with this message:

```
 %DELETE-W-FILNOTDEL, error deleting NEPDISK:[JOEUSER]URBASE_TEST.DIR;1
 -RMS-E-PRV, insufficient privilege or file protection violation
```

This means that Joe lacks the proper access to delete this subdirectory. This is not a big problem; because it's a subdirectory of [JOEUSER], his personal root directory, he can give himself the necessary permission with the SET PROTECTION command. Section 9.1.8, "Controlling Access to a File," explains this more fully; for now, it's enough to know that Joe only needs to type the following to give himself permission to delete the URBASE_TEST.DIR "file":

```
set protection=(o:d) urbase_test.dir;1
```

The "o" stands for "owner," and the "d" for "delete." Although there are other categories of users and other categories of access to files, at this point Joe only cares about giving the owner (himself) permission to delete it. All other privileges for all other users are irrelevant, because soon `URBASE_TEST.DIR` will no longer exist.

After he resets the protection level, he can delete the file without any trouble.

If you try to delete a subdirectory that has any files in it, VMS responds like this:

```
%DELETE-W-FILNOTDEL, error deleting NEPDISK:[JOEUSER]URBASE_TEST.DIR;
1-RMS-E-MKD, ACP could not mark file for deletion
-SYSTEM-F-DIRNOTEMPTY, directory file is not empty
```

The solution is simple enough: delete anything in the subdirectory, and enter the `DELETE` command again.

Chapter 10 The OpenVMS EVE Text Editor

VMS offers several built-in text editors. Entering `EDIT` at the DCL prompt, with no qualifiers, invokes the EDT line editor. This was the most popular VMS text editor for a long time. Because it's a line editor, using it means entering commands to go in and out of command, edit, and insert modes. When you edit with the EDT editor, you edit one line at a time. Although it does have a full-screen mode, where you display your file and move your cursor around to edit wherever you want, you don't need to bother with EDT because of the superior full-screen editing offered by a more recent VMS editor called the Extensible VAX Editor, or EVE.

EVE could very well be the easiest text editor to learn and use on any minicomputer or mainframe. The commands are simple without limiting you too much, and EVE provides its own command language that makes it easy to customize and add features to the editor.

The vi Text Editor on VMS

Some VMS systems have the UNIX vi editor installed, so many UNIX users don't need to learn any of the VMS editors. (Unless you're a real vi expert, you should at least check out EVE—a vi beginner will almost certainly prefer it.) If you do look for vi on a VMS system, be aware of a VMS utility called VIEW used for looking at certain kinds of documents. If you try to start up vi and it's not on the system, you may start up VIEW instead. It will probably just give you an error message and return you to the DCL prompt, because VIEW expects very specific types of files as input.

10.1 EVE and Special Keys

On the DEC VT220 terminal and its more advanced successors, you perform many of the most important editing functions by using special keys with their names written right on them. Examples include the Help, Do, and Find keys. If you use a PC running a program to emulate one of

these terminals, chances are that certain function keys will do the job of these specialized keys.

You can also use the numeric keypad on the right of your keyboard to accomplish many of EVE's more advanced tricks. In certain situations, you can use a key known as the "Gold" key (usually the key in the numeric keypad's upper-left corner—PF1 on a VT terminal, or the Num-Lock key on a PC emulating one) in combination with others, similar to the way you use the Control key in combination with letter, number, and function keys.

You don't need these keys, in their Gold version or otherwise, to accomplish the most basic editing tasks; a couple of the specialized keys mentioned above (or their surrogate function keys) provide all you need.

On a PC, the only function key that you have to make sure to remember is F2— the Help key. As you'll see in section 10.9, "EVE On-line Help," the Help key shows you what the other keys do. After you've used EVE a little, you'll no longer need F2 to remind you about F4, because you'll use it often: it's the PC's substitute for the "Do" key, the key that displays the `Command:` prompt below the EVE status line. Commands at this prompt enable you to accomplish basic tasks such as saving your work and quitting EVE.

10.2 Entering EVE

The `EDIT` command can start up several different VMS editors; you indicate which you want by the qualifier you use. Because `/EDT` is the default, entering `EDIT` without any qualifier enters the line editor. (If you accidentally enter the `EDIT/EDT` editor, enter `QUIT` at the `*` prompt to get out of it.)

Unfortunately, you don't enter `EVE` by entering `EVE` or `EDIT/EVE`, so you must remember its other name: the Text Processing Utility, or TPU. (Theoretically, the TPU could be configured to run other editors besides EVE, so I don't mean to make it sound as if TPU and EVE mean the same thing.) When you enter `EDIT/TPU` at the DCL command prompt followed by a file name, the editor displays that file for you to edit if it exists or creates a new file with that name if it doesn't. For example, when Joe User enters

```
edit/tpu rochester.txt
```

and presses Return, the TPU starts up EVE, and—assuming that Joe has no file called `rochester.txt`—displays a message (see the bottom of Figure 10.1) informing him that because none exists, it is creating `rochester.txt`.

```
[End of file]
```

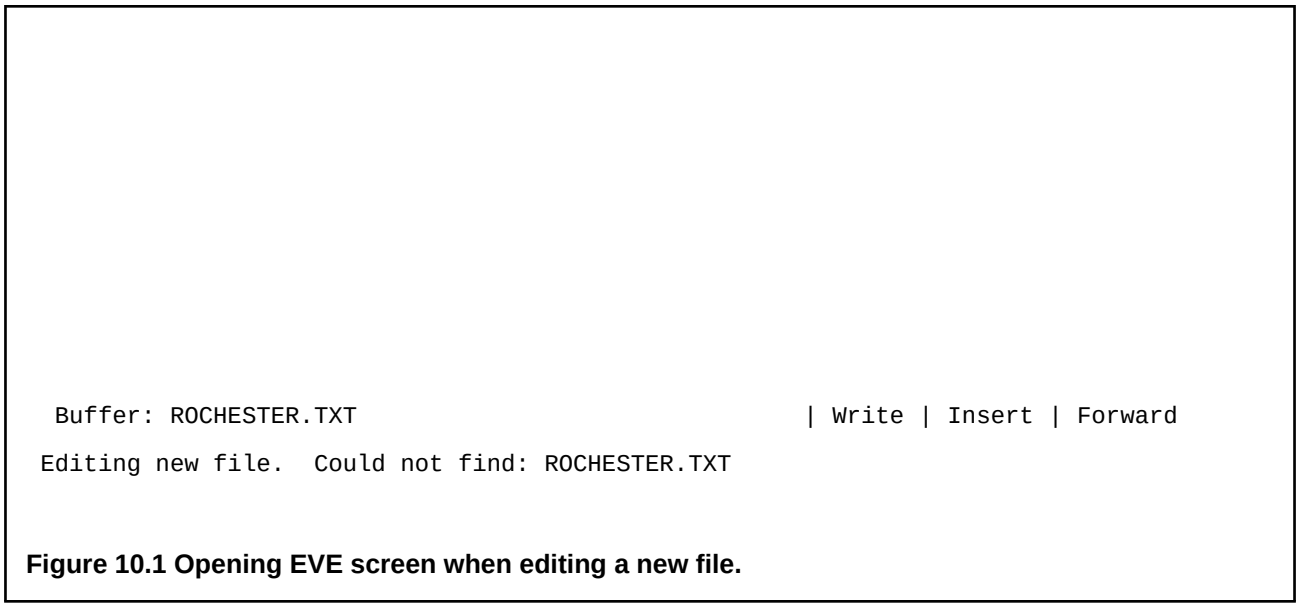


Figure 10.1 Opening EVE screen when editing a new file.

If the file had existed, and been six lines long, the message at the bottom would have read:

```
6 lines read from file NEPDISK:[JOEUSER]ROCHESTER.TXT;1
```

Because the new file is empty before you enter anything, the editor's "End of file" message appears at the top of the screen. The status line at the screen's bottom tells you three things:

- You can write to this file. In other words, you can make changes to it and save those changes. Sometimes people use EVE to just look at a file with no intention of changing it; adding the `/NOWRITE` qualifier to `EDIT/TPU` brings up the file with a status line similar to this:

  ```
  Buffer: ROCHESTER.TXT | Read-only | Unmodifiable | Forward
  ```

  When doing this, you can only read the file, and the `Insert` part of the status line becomes irrelevant, because the file is "Unmodifiable."

- When you are in "Write" mode, the "Insert" part shows that EVE will insert newly entered text at the cursor's position instead of overlaying existing text. As you'll see in section 10.5, "Typing Over Existing Text," Ctrl+A toggles between Insert and Overstrike mode.

- The final message on the status line shows that the default direction for commands that scroll the text is Forward. EVE doesn't give you separate commands to move or search

backwards and forwards; you have commands to scroll text a certain amount, and a command to search for text. The direction in which these commands move through your file depends on the current default direction. Section 10.6, "Searching for Text," shows how to change direction.

10.3 Inserting Text

As the bottom of the screen shows, you begin in Insert mode. You don't have to enter or press any special keys to let EVE know that you want to insert text; you can just start typing. Press Return to move the cursor to a new line. To insert a new line between two existing lines, use your cursor keys to move your cursor to the beginning of the line after the insertion point and press Return.

As you type in text, the "End of File" message moves down to show the current position of the end of the file, as shown in Figure 10.2.

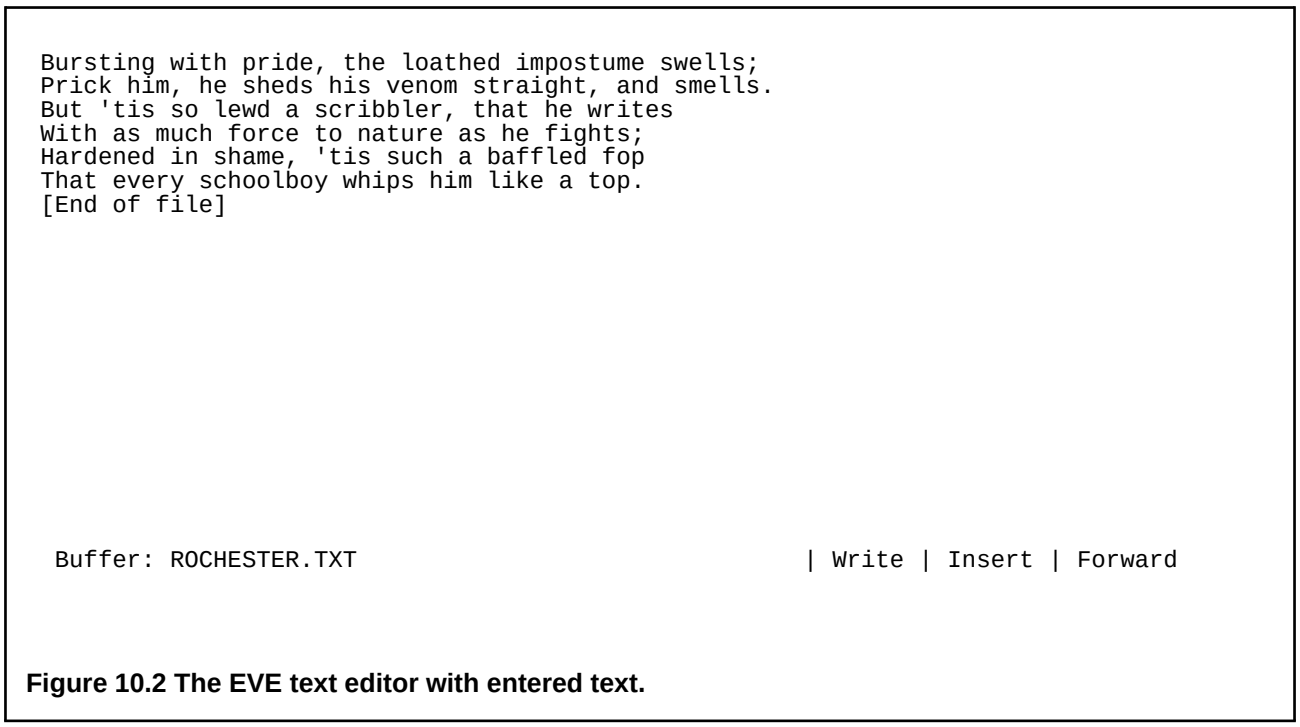


Figure 10.2 The EVE text editor with entered text.

10.4 Deleting Text

Pressing the Delete key has the same effect in EVE as it has on the DCL command line: it deletes the character immediately to the left of the cursor's current location. In Insert mode, it shifts all characters on the right of the deleted character to the left when it deletes it. In Overstrike mode, it acts more like a space bar that sends your cursor to the left: it replaces the deleted character with a space and leaves the other characters on that line alone.

To delete a blank line, make sure you're in Insert mode (see section 10.5, "Typing Over Existing Text," for more on the Insert and Overstrike modes), move to the beginning of the following line and press the Delete key.

10.5 Typing Over Existing Text

The status line at the bottom of the screen shows whether you are in Insert or Overstrike mode. To change from one to the other, press Ctrl+A. In Insert mode, any new characters typed at the cursor's location move the existing characters on the cursor's right one character further to the right. In Overstrike mode, new characters appear in place of the existing characters at the cursor.

10.6 Searching for Text

Pressing the Find key (on a PC, the F1 key) displays this prompt just below the EVE status line:

```
Forward Find:
```

(It might say "Reverse Find," depending on the current default scrolling direction, as indicated by your status line. To change the direction of searches, the Change Direction key is F11 on a VT terminal and F3 on a PC.) Enter the text to search for at the `Forward Find:` prompt and press Return. EVE searches for the entered text and highlights it if it finds it. If it cannot find the text you enter, it tells you so. For example, if you tell it to search for "potrzebie" and it doesn't find it, it tells you

```
Could not find: potrzebie
```

To repeat a search, press F1 twice. The first time you press it, you will see nothing at the `Forward Find:` or `Reverse Find:` prompt; the second time, EVE searches for the next occurrence of the last search target that you entered at that prompt.

You can control whether the search is case-sensitive by pressing the Do key (F4 on a PC) to display the `Command:` prompt and entering `SET FIND CASE EXACT` to make searches case-sensitive or `SET FIND CASE NOEXACT` to tell EVE to ignore the case of the text it searches.

10.7 Saving Your Changes

To save your file, press the Do key to display the `Command:` prompt, enter the `WRITE` com-

mand with no parameters, and press Return. EVE saves your file and returns your cursor to its former position so that you can continue to edit the file. (The `SAVE FILE` command also accomplishes this, but the many options of the `SAVE` command besides `FILE` make `WRITE` a simpler way to save your work.) Section 10.8, "Quitting EVE," shows how to save your work and and quit EVE in one command.

10.8 Quitting EVE

Enter `QUIT` at the Do key's `Command:` prompt to quit EVE and return to the DCL prompt. If you've made changes to your file without saving them, EVE displays the following:

```
Buffer modifications will not be saved, continue quitting [Yes]?
```

The square brackets show that "Yes" is the default answer. In other words, if you press Return without entering anything, EVE considers that a "Yes" and finishes the quitting process. This sends you back to the DCL prompt without saving the changes you've made since the last time you saved. Entering "N" for "No" at the `Buffer modifications` prompt tells EVE that you changed your mind about quitting. The prompt goes away, and EVE returns your cursor to its location before you pressed the Do key so that you can continue to edit the file.

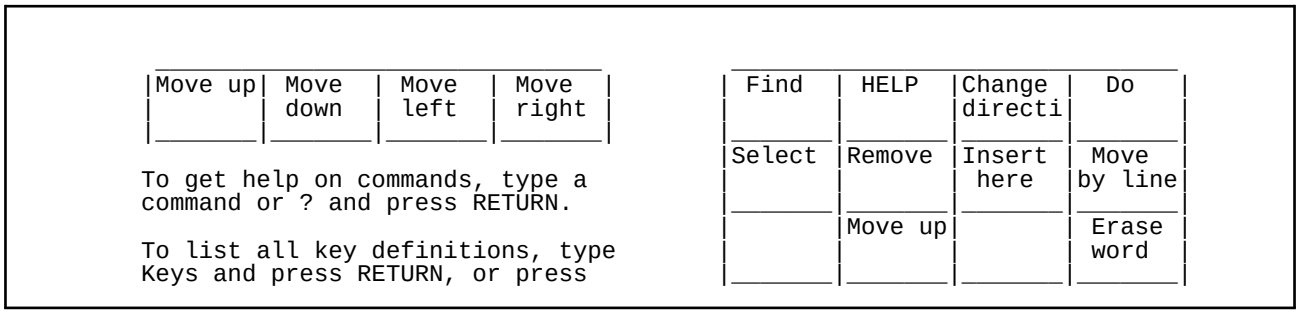
Enter the `EXIT` command at the Do key's `Command:` prompt to save your work and quit EVE all at once.

10.9 EVE On-line Help

EVE offers two ways to get help:

- Pressing the Help key (or on a PC emulating a terminal, F2) displays help about the use of the keyboard in EVE.
- Entering `HELP` at the Do key's `Command:` prompt displays the EVE general help menu.

Figure 10.3 shows a typical opening help screen after you press the Help key while using EVE (the actual screen you see depends on the terminal you use).



```
       GOLD-HELP.                                    | Move  | Move  | Move  |       |
                                                     | left  | down  | right |Change |
       To show a key definition, use                 |_______|_______|_______| mode  |
       SHOW KEY.                                     |  Next screen  |Previou|       |
                                                     |               |screen |       |
                                                     |_______________|_______|_______|

                                                        Synonyms for the DO key:
                                                                  PF4

 Buffer: HELP
Press the key that you want help on (RETURN to exit help):
```

Figure 10.3 Opening EVE help screen.

As the bottom of the screen tells you, you can display help about a specific key by pressing that key while viewing this screen. For example, press F1 (the Find key—as the diagram shows, this is the upper-left corner of the numeric keypad, which is NumLock on a PC and PF1 on an actual VT terminal) to display help about searching for text, as shown in Figure 10.4.

```
  FIND

   Searches for a string of text you specify and highlights the found text.

   Keys:  EVE Default    VT100 Keypad    EDT Keypad    WPS Keypad
          ---------------------------------------------------------
          FIND           PF1             GOLD-PF3      GOLD-, on keyboard

   Steps:

      1.  Use FIND (see key list above).

      2.  Type the text you want to find (see examples below).

      3.  To find another occurrence of the same string,
          use FIND NEXT or press the FIND key twice.

   Examples:

      Commands               Purposes or Effects
      ------------------------------------------------------------------
 Buffer: HELP        To see more, use:Press Up arrow or Down arrow to see more.
Press the key that you want help on (HELP for keypad, Return to exit help):
```

Figure 10.4 Help screen for EVE's Find key.

The bottom of the screen tells you which keys scroll through this help information. On a VT terminal, these are the Prev Scrn and Next Scrn keys; when using a terminal emulation program, the Cursor Up and Cursor Down keys will probably scroll through the help information.

Pressing Return without entering anything leaves the help facility and returns to the file you were editing. Pressing any other special key displays similar help about that key. Entering a question mark (?) displays the same main help screen available when you enter `HELP` at the Do key's `Command:` prompt.

Enter `HELP` at the Do key's `Command:` prompt to display the general help menu. Figure 10.5 shows the beginning of this help.

```
   List Of Topics (Commands)

    For help on EVE topics, type the name of a topic and press RETURN.

       o  For a keypad diagram, press HELP.
       o  For help on VAXTPU builtins, type TPU and press RETURN.
       o  To exit from help and resume editing, press RETURN.

    EDITING TEXT

       Change Mode              Erase Word            Restore Character
       Copy                     Insert Here           Restore Line
       Cut                      Insert Mode           Restore Selection
       Delete                   Overstrike Mode       Restore Sentence
       Erase Character          Paste                 Restore Word
       Erase Line               Quote                 Select
       Erase Previous Word      Remove                Select All
       Erase Start Of Line      Restore               Store Text

    BOX OPERATIONS

  Buffer: HELP        To see more, use:Press Up arrow or Down arrow to see more.
 Type the topic you want help on (press RETURN if done):
```

Figure 10.5 EVE's general help menu.

At the `Type the topic you want help on` prompt, enter any of the words or phrases listed on the screen, such as `Delete` or `Erase Character`. Remember, these choices are merely the beginning; just as EDITING TEXT is a heading for a series of choices, so are BOX OPERATIONS and several other categories. Scrolling down displays the rest.

The EVE help works essentially the same as regular VMS help. If you choose a broad enough

topic, its help text will end with subtopics from which to choose.

10.10 Other EVE Features

EVE has just about all the capabilities that you could ever want in a text editor or word processor. This includes:

- The ability to extend TPU by writing command files.
- The ability to mark ranges and blocks of text for moving, copying, and deletion.
- The ability to split the screen into multiple windows so that you can edit multiple files at once.

A description of how to use these features is beyond the scope of this chapter, but you can find out more about them easily enough by exploring EVE's on-line help.

Chapter 11 Using an OpenVMS System

11.1 Printing Text Files

Printing in VMS is simple. Enter the PRINT command followed by the name of the file to print. For example, if Joe User enters

```
PRINT ROCHESTER.TXT
```

he might see a response from the system like this:

```
Job ROCHESTER (queue SYS$PRINT,entry 203) started on SYS$PRINT
```

SYS$PRINT is the name of the default print queue; if Joe had wanted to print ROCHESTER.TXT on a different printer, he would add the /QUEUE=queuename qualifier to the end of his print command.

If other print jobs are ahead of this one in the print queue so that it must wait its turn to start printing, the started on SYSPRINT line will say pending.

11.1.1 Checking the Print Queue

To list the jobs waiting to print, enter the SHOW QUEUE command followed by the name of the print queue:

```
SHOW QUEUE SYS$PRINT
```

The output lists the file currently printing, as well as any waiting files:

```
 Printer queue SYS$PRINT, on NEPTUNE::LCA0:, mounted form DEFAULT
   Entry  Jobname         Username  Blocks  Status
   -----  -------         --------  ------  ------
     588  091394JS        MJONES         2  Printing
     590  ROCHESTER       JOEUSER        1  Pending
     591  TESTDCL         MJONES         1  Pending
```

The columns of information are self-explanatory, with one exception: Entry. This is how VMS identifies individual print jobs, and how you must refer to your print job if you want to cancel it.

11.1.2 Cancelling Your Print Job

To cancel a pending print job, you use the same DELETE command that you use to delete files and subdirectories, except that you add the /ENTRY=entrynumber qualifier. The entry number is the number that showed up in the first column when you entered SHOW QUEUE SYS$PRINT. For example, Joe enters the following to delete ROCHESTER.TXT from the list of files shown above:

```
DELETE/ENTRY=590
```

Since VMS displays no confirmation of a successful deletion from the print queue, you must enter the SHOW QUEUE command again to see if the job you deleted is gone. If you didn't get an error message when you entered the DELETE/ENTRY command, chances are that the file was successfully deleted. The following is the most common error message:

```
 %DELETE-E-NOTDELETED, error deleting 590
 -JBC-E-NOSUCHJOB, no such job
```

In other words, it couldn't find entry 590. Either the number was mistyped or VMS has already printed that particular job. Either way, the queue has no job with that number waiting to print.

An optional parameter for the DELETE/ENTRY command is the name of the print queue. If you omit it, VMS assumes that you mean the SYS$PRINT queue. If Joe had sent the ROCHESTER.TXT file to a queue called ACCTING, he would have checked out the queue with the following command:

```
SHOW QUEUE ACCTING
```

If ROCHESTER.TXT had an entry number of 304, he would delete it from that queue with this command:

```
DELETE/ENTRY=304 ACCTING
```

11.2 Command Files

On most operating systems, one-line command files are the best way to make it easier to enter long but commonly used commands. Although DCL command files are useful and easy to create, VMS has an even better way to define a short string of characters to represent a much longer one: symbols.

11.2.1 Symbols

Defining a symbol assigns one string of characters to represent another. For example, if you like to use the /DATE/VER=1/SIZE qualifiers with the DIRECTORY command, but get tired of repeatedly typing

```
DIR/DATE/VER=1/SIZE
```

every time you want to list your file names, you can assign this command to a symbol. If the symbol is the string FILES, you would assign it like this:

```
FILES=="DIR/DATE/VER=1/SIZE"
```

Entering FILES at the DCL command prompt would then have the same effect as entering DIR/DATE/VER=1/SIZE.

When you create the symbol, don't forget the quotes around the string being assigned to the symbol. You don't always need both equal signs when creating a symbol at the DCL command prompt, but two are necessary when you put a symbol assignment command in a command procedure, so it's a good idea to get into the habit of using two.

A symbol assignment only lasts until you log out. If you want certain symbol assignments done every time you log in (and everyone has a few favorites), add the commands that create the symbols to your `LOGIN.COM` command procedure. (For more on `LOGIN.COM`, see section 11.2.3, "The Automatic Login Command File.")

11.2.2 DCL Command Procedures

A DCL command procedure is a file that consists of DCL commands. Each line begins with a dollar sign ($) and the last line is the following:

```
$ EXIT
```

Command files have a file type of `COM`, and you start them up by typing the file name preceded by an "at" sign (@).

For example, let's say Joe User has a program called `SUMRPT` that creates and stores a report in a text file called `SUMRPT.TXT`. Every time he runs this report he examines the report output with the EVE text editor, perhaps changing a few things, and then prints it. He could automate these steps by putting them in a file called `SUM.COM` with these lines:

```
! SUM.COM: run summary report, EVE it, print it. 2/13/94 J. User
$ SUMRPT
$ EDIT/TPU SUMRPT.TXT
$ PRINT SUMRPT.TXT
$ SHOW QUEUE SYS$PRINT
$ EXIT
```

The first line begins with an exclamation point. This tells VMS to ignore the line; it is a comment that describes the purpose of the program. Complicated command procedures need many comment lines so that someone who reads them at a later date can easily see the purpose of each part.

Except for the dollar signs, the next four lines show just what Joe would have typed at the DCL prompt to execute these steps. (The `SHOW QUEUE SYS$PRINT` line is an added bonus that he threw in because most people want to know immediately how long they have to wait for something that they sent to the printer.) The final line, `$ EXIT`, returns control to the DCL prompt.

To run this command procedure, Joe types

```
@SUM
```

at the DCL prompt. VMS looks for `SUM.COM`, finds it, and executes its commands one by one.

(Actually, VMS first checks to make sure that there are no DCL commands with this name. This is why you should never name a COM file after an existing command, like `COPY.COM` or `PRINT.COM`.)

DCL command procedures can be much more complex than `SUM.COM`. It's a whole programming language, complete with branching, variables, and subroutines. Built-in functions known as lexical functions give DCL command procedures the ability to manipulate string expressions, convert between string and numeric data, query the operating system for information, and many other things more often associated with programming languages like C and Pascal than with an operating system command procedure language.

11.2.3 The Automatic Login Command File

When a user first logs in to a VMS system, the system looks for a command procedure called `LOGIN.COM`. If it's in the user's root directory, the system executes it. This is particularly useful for defining your favorite symbols that you want to use every time you log in. Joe User's `LOGIN.COM` command procedure shows how you can define symbols to substitute new words for often-used commands that may be hard to remember:

```
$ SHOW QUOTA
$ FILES=="DIR/DATE/VER=1/SIZE"
$ EVE=="EDIT/TPU"
$ SHOW=="TYPE/PAGE"
$ EXIT
```

Joe's `LOGIN.COM` file begins with the command `SHOW QUOTA`. This command, discussed in section 8.3.2, "Querying Available Disk Space," displays information that may be handy when you first log in.

The next three lines create symbols for Joe to use as shortcuts in his VMS session. For example, after his `LOGIN.COM` command file executes, Joe can type

```
eve schedule.txt
```

and VMS reacts as if he had typed this:

```
edit/tpu schedule.txt
```

It also substitutes "TYPE/PAGE" everywhere that he types `SHOW` at the DCL prompt and "DIR/DATE/VER=1/SIZE" whenever he types `FILES`.

11.3 Communicating with Other Users

The VMS `MAIL` program has its own command line, commands, and on-line help to aid you in creating, sending, receiving, and managing mail messages. It's one of the best mail programs built-in to any operating system.

To start the program, enter the word MAIL at the DCL prompt. VMS tells you whether you have new messages waiting and displays the MAIL program's prompt:

```
You have 1 new message.
MAIL>
```

To send a mail message, enter the command SEND and press Return. The MAIL program prompts you for the name of the message's recipient:

```
MAIL> send
To:
```

Enter the recipient's login ID and press Return. If the system doesn't recognize the name (for example, MJOONES), it tells you this:

```
%MAIL-E-NOSUCHUSR, no such user MJOONES
```

If it does recognize the recipient's name, it prompts you for the subject of the message. Enter the subject, press Return, and the mail program displays instructions about entering your message:

```
MAIL> send
To:     mjones
Subj:   budget meeting
Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:
_
```

The cursor appears under the instructions, ready for your input. You are only allowed the most primitive kind of input; each time you press Return at the end of a line, you cannot go back and edit that line. We'll see shortly how to send an existing text file, which gives you the flexibility to compose your mail message with the EVE text editor before you send it.

As the explanatory message tells you, press Ctrl+C to abort your message or Ctrl+Z when you are satisfied with it. If you abort, the MAIL program acknowledges your action with the word "Cancel" and this message:

```
%MAIL-E-SENDABORT, no message sent
```

If you press Ctrl+Z to send your message, it displays the word "Exit."

11.3.1 Sending an Existing File

If you add a parameter to the SEND command described earlier, the MAIL program assumes that it names a file that you want to send. For example, if Joe enters

```
send marymemo.txt
```

at the MAIL> prompt, the MAIL program prompts him for the recipient's name and the message's subject, as usual, but it then returns him to the MAIL> prompt. If MARYMEMO.TXT doesn't exist, it displays an error message.

11.3.2 Receiving Mail

If you are logged in when someone sends you a mail message, VMS displays a message similar to this on your screen:

```
New mail on node NEPTUNE from OROURKE::FDAKOVA     (12:55:02)
```

If you are not logged in, your system probably displays a message that tells you how many unread messages are waiting for you the next time you log in. As we saw in section 11.3, "Communicating with Other Users," the `MAIL` program also tells you how many new messages you have when you start it up.

11.3.2.1 Mail Folders

The `MAIL` program organizes messages into groups called folders. It's easier to learn how to read, delete, save, and organize mail messages if you first understand the role of folders.

Creating folders and moving messages between them is easy. `MAIL` automatically creates three folders for you as you need them: NEWMAIL, MAIL, and WASTEBASKET. To avoid confusion, keep in mind that certain actions automatically move messages in and out of these three folders.

VMS stores new, unread messages in the NEWMAIL folder. When you leave the NEWMAIL folder, the mail program automatically moves any newly read messages from there to the MAIL folder if you didn't explicitly move them to any other folders after you read them. Deleting a message from any folder moves it to the WASTEBASKET folder.

At any given time, one folder is the current one. For example, when you start up `MAIL`, the NEWMAIL folder is current if you have any unread messages. If you do not have any unread messages, the MAIL folder is the current one.

To make a different folder current, enter the `SELECT` command at the `MAIL>` prompt followed by the name of the folder you want to make current. For example,

```
SELECT WASTEBASKET
```

makes the WASTEBASKET folder the current one.

Use the `DIR` command to list the messages in a given folder. If you don't include a folder name, the `MAIL` program assumes that you want to list the messages in the current folder. For example, if NEWMAIL is your current folder, entering

```
DIR
```

lists the messages in the NEWMAIL folder, and

```
DIR MAIL
```

lists the messages in the MAIL folder. A message list looks something like this:

```
 MAIL    # From                 Date         Subject
     1 NEPTUNE::MJONES      17-AUG-1994  1995 budget
     2 OROURKE::FDAKOVA     18-AUG-1994  new Windows release?
     3 NEPTUNE::KBERRY      19-AUG-1994  lunch Monday
     4 NEPTUNE::LSTORCH     19-AUG-1994  Giants tickets
```

To list folder names instead of information about messages, add the `/FOLDER` qualifier to the `DIR` command.

Because the `MAIL` program creates the NEWMAIL and WASTEBASKET folders as you need them, they may not always exist. If you enter the `MAIL` program, but have no new mail, and you enter the command

```
SELECT NEWMAIL
```

you'll see an error message like this:

```
%MAIL-E-NOTEXIST, folder NEWMAIL does not exist
```

If you have no new messages, you don't have a NEWMAIL folder. A similar idea applies to WASTEBASKET: after you start up the `MAIL` program, you won't have a WASTEBASKET folder until you delete your first message (unless you last left the `MAIL` program with `QUIT` instead of `EXIT`—see section 11.3.2.5, "Leaving the MAIL Program," for more on this).

Section 11.3.2.2, "Reading Mail," shows how to move messages from one folder to another.

11.3.2.2 Reading Mail

Entering `READ` displays the first page of the oldest message in the current folder. If you have any new messages when you first start up the `MAIL` program, your NEWMAIL folder will be current, so `MAIL` will display your oldest unread message. If the message you are reading is more than one page long, press Return to advance to the next page.

As a matter of fact, since `READ` is the default command in the `MAIL` program, you only need to press Return to read the next message in the current folder. To read a different message in the current folder, enter its number, which you can learn by entering `DIR` to list the folder's messages.

11.3.2.3 Moving a Message to Another Folder

The `FILE` command (and the `MOVE` command, which behaves identically) tells the `MAIL` program to move a message to another folder. To move the current message, enter `FILE` at the `MAIL>` prompt and the `MAIL` program prompts you for the name of the destination folder. If you enter the name of a nonexistent folder, `MAIL` asks you if you want to create a folder with that

name.

For example, after he reads the following message, Joe User enters `file budget95` at the `MAIL>` prompt. The `MAIL` program responds by telling him that no such folder exists, and asks if he wants to create one:

```
NEWMAIL
From:   NEPTUNE::MJONES
To:     JOEUSER
CC:
Subj:   1995 budget
I know it seems early, but it's already time to talk about the
budget for fiscal 1995.  Get together any notes or ideas you have
and get in touch with me.  I'm out of the office on Tuesday.
MAIL> file budget95
Folder BUDGET95 does not exist.
Do you want to create it (Y/N, default is N)?
```

Joe responds with a "Y" for "Yes," and the `MAIL` program creates the BUDGET95 folder and stores this message there.

Just as adding the first message to a folder creates that folder, moving the last message from a folder automatically deletes that folder. The `MAIL` program has no explicit command for deleting a folder.

Any messages that you read in the NEWMAIL folder without deleting or filing to another folder get automatically moved to the MAIL folder when you leave NEWMAIL. This happens whether you leave NEWMAIL by selecting another folder or by exiting the `MAIL` program. If this includes all the messages in NEWMAIL, then the folder itself will be deleted until the next time you start up the `MAIL` program and have new mail.

11.3.2.4 Deleting Messages from a Folder

Entering `DELETE` by itself at the `MAIL>` prompt either deletes the message you are currently reading, if it takes up more than one screen and is pausing for you, or the message you just read, if you are between messages. You can also add one or more numbers to the `DELETE` command to specify which of the messages in the current folder to delete. For example, if your current folder has the messages shown in section 11.3.2.1 earlier ("Mail Folders") and you enter the following command

```
DELETE 2,4
```

and follow it with the `DIR` command, the `MAIL` program will show the following revised list:

```
MAIL    # From                Date         Subject
    1 NEPTUNE::MJONES      17-AUG-1994  1995 budget
    2 (Deleted)
    3 NEPTUNE::KBERRY      19-AUG-1994  lunch Monday
    4 (Deleted)
```

To undelete a message, select the WASTEBASKET folder and move the message in question to another folder with the `FILE` or `MOVE` command. It doesn't have to be moved to the folder

where it was originally located at the time you deleted it.

11.3.2.5 Leaving the MAIL Program

The MAIL program offers two ways to exit and return to the DCL command prompt:

- The QUIT command leaves anything in your WASTEBASKET folder alone when it returns to the DCL prompt. The next time you enter the MAIL program, your WASTEBASKET folder will still be there with everything that you deleted in your previous session.
- The EXIT command empties out your WASTEBASKET folder when it returns you to the DCL prompt.

11.3.2.6 Saving a Message in a Text File

The EXTRACT command saves the message that you read most recently (or are currently reading) into a text file in the default directory. In the following example, Joe saves the message from Mary in a file called 121794MJ.TXT:

```
MAIL> extract 121794mj.txt
%MAIL-I-CREATED, NEPDISK:[JOEUSER]121794MJ.TXT;1 created
```

The system responds with a message that it has successfully created the file.

11.3.3 On-line Help in the MAIL Program

On-line help in the MAIL program resembles on-line help elsewhere in VMS. Enter HELP by itself at the MAIL> prompt to display an introduction to using help and a list of commands that you can learn more about. This list includes all the MAIL commands described in this section and several more.

As with help from the DCL prompt, if you know the command that you want to learn more about, you can enter that command name as a parameter to the HELP command at the MAIL> prompt. For example, entering

```
HELP SELECT
```

tells you about the command to make a particular folder current.

11.4 A Sample OpenVMS Session

One morning you arrive at your desk, log in to your VMS account, and see the following line along with the other login messages:

```
You have 1 new Mail message.
```

You enter MAIL at the DCL prompt to start the mail program, and the MAIL> prompt appears. Because you know that pressing Return has the same effect as entering the MAIL program's READ command and that without additional instructions it will show you the oldest unread message in your NEWMAIL folder, you press Return and this message shows up on your screen:

```
NEWMAIL
From:   NEPTUNE::LNIVEN
To:     JOEUSER
CC:
Subj:   August 1994 numbers
I need the gross and net figures, by region, for last month.
Please run the report and send me the output by e-mail before
lunch.  I need them for a 2PM meeting.
Thanks!
MAIL>
```

Since you know that the data Larry needs isn't complete, you decide to keep a disk file copy of this memo in case the unreadiness of the numbers comes back to haunt you. You enter EXTRACT NIVMEMO.TXT at the MAIL> prompt, and then EXIT to return to the DCL prompt.

Because this report must be run every month, you've already set up a command file called SUMRPT to run the report and save the data in a file named SUMRPT.TXT. You run the command file:

```
@SUMRPT
```

When the command file finishes, you take a quick look at the report output by using the TYPE command to display the SUMRPT.TXT file that was just created:

```
$ type sumrpt.txt
```

The system responds by showing you SUMRPT.TXT, as shown in Figure 11.1.

```
AUGUST   1994           Gross     Net
                       ------  ------
Northeast              75,732  11,890
Mid-Atlantic           69,348  11,008
Southeast              61,835   8,890
Southwest                   0       0
Midwest                70,762  12,934
Northwest              14,242   1,634
West Coast                  0       0
                       ------  ------
AUGUST   TOTAL        291,919  26,356
```

Figure 11.1 The contents of the SUMRPT.TXT file.

It looks like the figures aren't in yet from the southwest and the west coast regions, so the totals aren't that useful. However, Larry asked for them, so Larry gets them. You pull up SUMRPT.TXT in the EVE text editor to add a note:

```
EDIT/TPU SUMRPT.TXT
```

Once EVE displays the file, you add a few lines at the top, so that your screen looks like Figure 11.2.

```
Larry -

As you can see, we're still waiting for the figures from the southwest
and from the west coast.  I left voice mail with Ginny in Phoenix and
Jim in LA; I'll call you when I know more.


AUGUST   1994           Gross     Net
                       ------  ------
Northeast              75,732  11,890
Mid-Atlantic           69,348  11,008
Southeast              61,835   8,890
Southwest                   0       0
Midwest                70,762  12,934
Northwest              14,242   1,634
West Coast                  0       0
                       ------  ------
AUGUST   TOTAL        291,919  26,356
[End of file]

 Buffer: SUMRPT.TXT                                    | Write | Insert | Forward

14 lines read from file NEPDISK:[JOEUSER]SUMRPT.TXT;4
```

Figure 11.2 Summary report file in EVE after editing.

After you add these new lines, you press F4 to display EVE's Command: prompt and type EXIT there to save your edits and return to the DCL prompt.

Now that the file is ready, you can send it to Larry. You enter MAIL at the DCL prompt to start up the mail program and SEND SUMRPT.TXT at the MAIL program's MAIL> prompt. When the mail program asks you for the user ID of the message's recipient, you enter LNIVEN. After you enter "August numbers" in response to the MAIL program's prompt for the subject of the message, the program returns you to its MAIL> prompt. You enter EXIT to return to the DCL prompt, finally entering LOGOUT at the DCL prompt to finish your VMS session.